Improving the Reliability of Skewed Caches through ECC based Hashes

MEDIAN 2014
March 28th, 2014 – Dresden, Germany

Improving the Reliability of Skewed Caches through ECC based Hashes

S. Yegin, B. Karsli, O. Ergin, M. Ottavi, S. Pontarelli, P. Reviriego

Received: date / Accepted: date

Abstract Skewed caches have been proposed to reduce the miss ratio on a cache by indexing the cache lines of a two way set associative cache not with the index but with a hash function of the index and the tag. This paper proposes to use ECC codes as the hashing functions that allow also correcting the stored data. It is shown that ECC behave very well as hash functions and that ECC protection provides gains in terms of error coverage with respect to state of the art skewed cache approaches.

Keywords Error correction codes · skewed cache

1 Introduction

Soft errors have been an increasingly important design issue in contemporary microprocessors. As the technology scales, circuits become more vulnerable to soft errors due to smaller intermediate capacitance nodes and denser chip layouts. On-chip caches constitute an important portion of the die area and are a major component for which mitigation techniques have to be implemented. Because of the criticality of the reliability of data stored inside the caches, all of the stored information is commonly protected against soft errors through Error Correcting Codes (ECC) [1]. Skewed caches are proposed as an alternative to set-associate cache design in order to improve the hit rate [2]. This is accomplished by separating the index values of different cache ways and using different hashing functions to calculate them. A hash function is any algorithm that takes an input of arbitrary length (called a key) and maps it to a value of fixed length (called a hash value or hash). Although a skewed cache structure offers better performance, it increases the size of the cache as it mandates the use of wider tag values and hence increases the vulnerability of the cache structure to soft errors. In this paper, we propose using an ECC as a hash function in a skewed cache design in order to achieve fault tolerance using less silicon area. We show that it is possible to have the performance of skewed cache design by using the same bit area as a regular set-associative cache.

2 Set Associative Cache versus Skewed Cache Design

Regular set associative cache design is shown in Fig. 1. The index bits are used to locate the entry where the corresponding data may be residing and the tag values are compared against each other to determine if there is a cache hit. Note that in both ways of the cache, the corresponding address locations reside on the same line and are indexed by the same bits inside the address. Also the tag values, as well as the data, stored in the cache ways are protected by ECC. Upon each access to the cache ways, ECC bits are checked to determine if there is a soft error on the stored bits. Skewed cache

This paper is part of a collaboration in the framework of COST ICT Action 1103 "Manufacturable and Dependable Multicore Architectures at Nanoscale."

This work was partially supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under the research grant 112E004.

S. Yegin B. Karsli and O. Ergin are with Department of Computer Engineering, TOBB University of Economics and Technology, Ankara, Turkey {sercan.yegin, burak.ibrahim.karsli}@gmail.com, oergin@etu.edu.tr
M. Ottavi and S. Pontarelli are with Department of Electronic Engineering, University of Rome Tor Vergata, Rome, Italy {ottavi,pontarelli}@ing.uniroma2.it
P. Reviriego is with Universidad Antonio de Nebrija, Calle Pirineos 55, 28040 Madrid, Spain previrie@nebrija.es

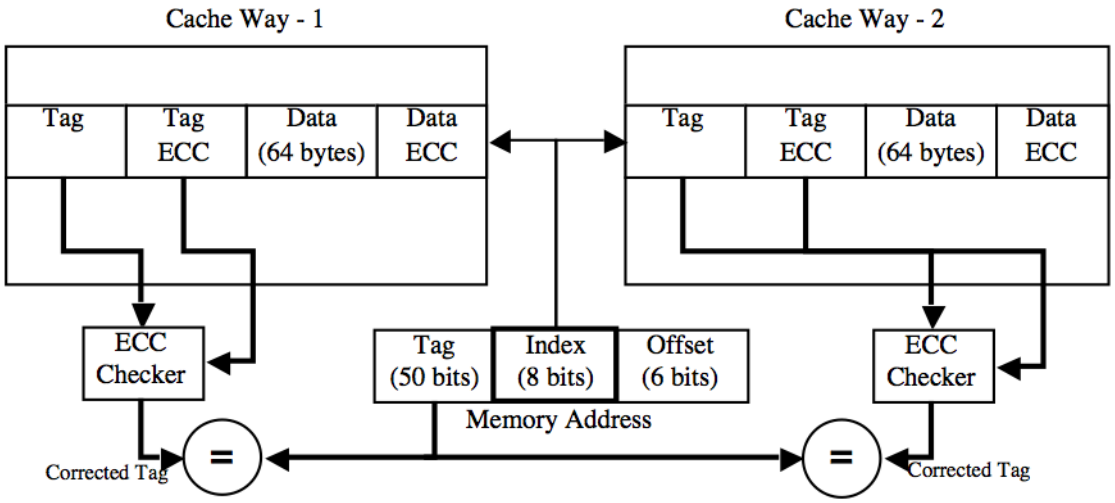

Fig. 1 Set Associative Cache Architecture

design is proposed as an alternative to set-associative cache (shown in Fig. 2). In the set-associative architecture, a location in the main memory can only be mapped to a single line inside the cache.

This may result in cache conflicts as in a 2-way architecture one location in main memory can only be mapped to two locations that reside on the same set, indexed by the same bits inside the address. In order to reduce the chances of conflicts inside the cache, cache ways are indexed separately in the skewed cache design using different hashing functions. As the results of the two hashing functions are different, the same address is mapped to two different physical entries inside each way. Skewed cache design effectively reduces the miss rate [2].

Skewed cache design does not use any part of the memory address for indexing; instead it uses the outcome of the hashing functions as index values to the cache ways. Because of this lack of indexing bits inside the memory address, the size of the stored cache tags are larger than the tags used in set-associative caches. Also the ECC bits are still stored inside the entries making it mandatory to use more bits inside each entry, when compared to the set-associative caches.

3 Proposed Scheme: Using ECC as the Hashing Function

Many different hashing functions can be used to map address to different locations in the cache ways. It has been previously shown that XOR mapping schemes achieve less miss rates when compared to the set-associative cache design [3]. Therefore we use this design choice as baseline in this paper. Since the ECC information is stored with each information area stored in the cache, values are encoded before they are written inside the storage space in order to generate the ECC bits. The Error Correcting Codes used for this purpose are themselves hashing functions that generate a bit vector from another input vector. As this hashing is done for fault detection purposes, we propose to use the ECC encoding circuit as the hashing function of the skewed cache and remove the stored ECC bits from the cache structure all together and use the ECC bits for indexing. This is similar to the scheme proposed to protect hash tables in [4]. Fig. 3 shows the proposed architecture where the computed ECC bits for the tags are no longer stored inside the cache ways but instead the ECC is used as the hashing function and the computed ECC bits are used as the indexes to the cache ways. Upon reading of the tag, the read value is checked for any possible soft errors before the stored tag is compared against the tag part of the memory address for a possible cache hit outcome. This new method, reduces the effective area of the cache and makes it less prone to soft errors as its silicon footprint is now smaller (it is less likely that a particle strike will occur as the area is now smaller). In the baseline cache (both in set associative and skewed) ECC bits are calculated before the tag value is first inserted inside the cache structure. In our implementation, since ECC values are not stored anymore, such a computation will not be performed. Instead the proposed architecture will both encode and decode tag bits in each access to the cache structure. The fact that the ECC encoding is now overlapped with the hashing stage means that the delay will be similar to the baseline skewed cache architecture. By using the proposed method we are effectively reducing the size of the cache to the level of a set associative cache while

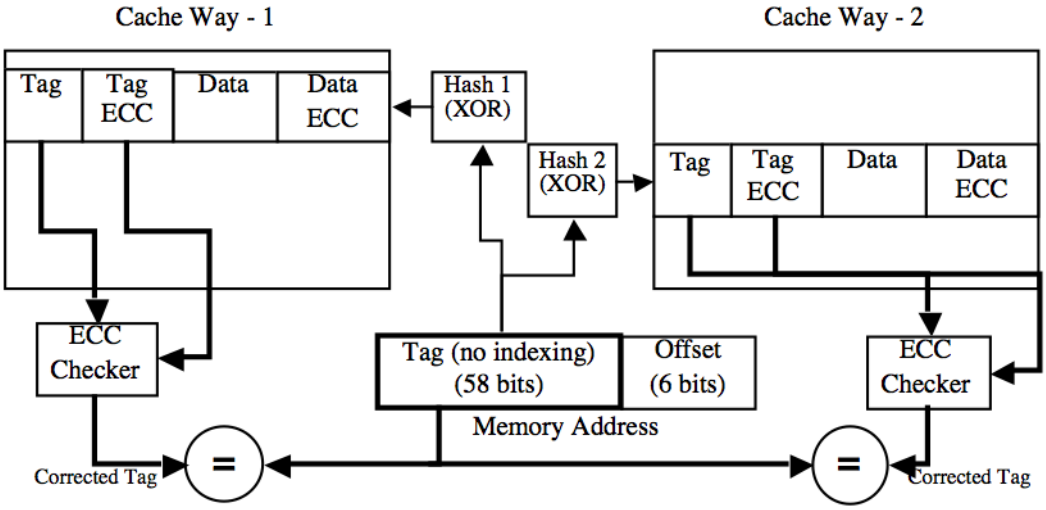

Fig. 2 Skewed Cache Architecture

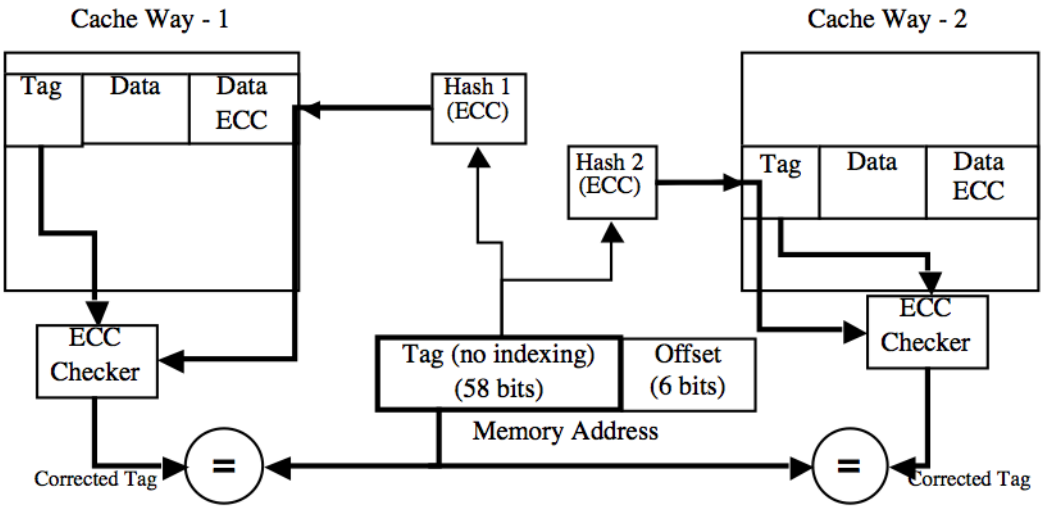

Fig. 3 Proposed Fault Tolerant Cache Architecture

keeping the performance benefits of the skewed cache design.

4 Evaluation Methodology

The proposed scheme reduces the area of the cache and therefore its vulnerability when compared to the baseline skewed cache architecture. In order to get an accurate idea about the performance, BCH ECC codes have been used as hashing functions [5] as opposed to the regular XOR hash functions. For the evaluation, we used MSIM [6], a cycle accurate microarchitectural simulator. In order to see the performance impact and avoid any shadowing effects of the level 1 cache on other levels of the cache. We applied the proposed technique on a 32 KB, 2 way cache with 64 bytes of data blocks and 2 cycles hit time. Spec 2006 benchmarks were run by fast forwarding 100 million and simulating 10 million instructions.

5 Results and Discussions

Table 1 shows the miss rates observed in the skewed cache with XOR hashing and the proposed BCH ECC hashing. Results show that with the exception of dealII, which shows poor miss rates, almost all benchmarks show cache miss rates close to skewed cache with XOR hashing, with omnetpp showing slightly better results. The savings in terms of cache area for this case study would be 8 bits per Tag as those are the bits needed to implement a SEC-DED code on a 58 bit block [1]. The overall savings will be approximately 1.4% of the cache area.

Table 1 L1 Cache miss rates for the baseline and the proposed scheme

Benchmark Name	Skewed Cache with XOR Hashing	Skewed Cache with ECC Hashing
400.perlbench	1.36%	1.36%
401.bzip2	0.08%	0.08%
416.gamess	0.00%	0.00%
429.mcf	0.20%	0.20%
433.milc	4.35%	4.35%
435.gromacs	0.07%	0.07%
436.cactusADM	20.19%	20.2%
444.namd	0.01%	0.01%
447.dealII	2.05%	3.27%
454.calculix	0.17%	0.15%
458.sjeng	12.50%	12.5%
462.libquantum	1.92%	1.92%
464.h264ref	0.00%	0.00%
470.lbm	0.00%	0.00%
471.omnetpp	0.18%	0.17%
473.astar	2.26%	2.26%
482.sphinx3	0.04%	0.04%
998.specrand	0.00%	0.00%
Average	2.52%	2.59%

6 Conclusions and Future Work

In this paper, we proposed an enhancement to the hashing functions of the skewed caches in order to improve soft error reliability. We show that by using the ECC as the hash function, it is possible to remove the need to store the ECC bits thus reducing significantly the area of the cache. Results from the spec 2006 benchmarks show that the proposed scheme achieves a similar performance to the skewed cache architecture. As future work, we want to find out how applicable is the proposed technique in the presence of multiple bit upsets. Furthermore, we expect to have similar results when we apply the proposed method on the higher cache levels.

References

1. C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: a state-of-the-art review", IBM Journal of Research and Development, vol. 28, no. 2, pp. 124-134, 1984.
2. Seznec, Andr. "A case for two-way skewed-associative caches." ACM SIGARCH Computer Architecture News. Vol. 21. No. 2. ACM, 1993.
3. Gonzlez, Antonio, et al. "Eliminating cache conflict misses through XOR-based placement functions." Proceedings of the 11th international conference on Supercomputing. ACM, 1997.
4. P. Reviriego, S. Pontarelli, J.A. Maestro, M. Ottavi, "Efficient Implementation of Error Correction Codes in Hash Tables", Microelectronics Reliability, 2013.
5. J.P. Grossman and L. Jakab "Using the BCH construction to generate robust linear hash functions", IEEE Information Theory Workshop, 2004.
6. Sharkey, J., "M-Sim: A Flexible, Multithreaded Architectural Simulation Environment", Technical Report CS-TR-05-DP01, Dept. of CS, SUNY - Binghamton, October 2005.